Splitting method for spatio-temporal search efforts planning

Mathieu Chouchane [a] Sébastien Paris [a] François Le Gland [b] Mustapha Ouladsine [a]

[a] *LSIS, Domaine universitaire de Saint-Jérôme, Avenue Escadrille Normandie Niemen, 13397 Marseille Cedex 20, France*

[b] *INRIA Rennes, Campus de Beaulieu, 263 avenue du Général Leclerc, 35042 Rennes Cedex, France*

Abstract

This article deals with the spatio-temporal sensors deployment in order to maximize detection probability of an intelligent and randomly moving target in an area under surveillance. Our work is based on the rare events simulation framework. More precisely, we derive a novel stochastic optimization algorithm based on the generalized splitting method. This new approach offers promising results without any state-space discretization and can handle various types of constraints.

Key words: Stochastic optimization; Sensor deployment; Rare events simulation; Generalized splitting; Search theory; Gibbs sampler; Markov chain Monte Carlo.

1 Introduction

Consider a randomly moving object that tries to cross a closed area or to run away from a known position. We want to compute the best spatial and temporal deployment of elementary search efforts in order to maximize the detection probability of this intelligent and randomly moving object. In our article, we call this object a *target* and the search efforts are assigned to *sensors*. Our work is closely linked to search theory [20], Stackelberg games [10] (which belong to game theory domain) and operational research. It is also related to sensor network [7].

Stackbelberg games [16] could have been used to model our problem, however, due to limitations it was not possible. In this two-person game, the first player, the *leader*, commits to a strategy first. Then, the second player, the *follower*, tries to optimize his reward, considering the action chosen by the leader. There are many security domains for which Stackbelberg games are appropriated. For example, the ARMOR system [15] has been used for years at the Los Angeles International Airport. ARMOR casts a patrolling and monitoring problem as a Bayesian Stackelberg game (an extension of Stackelberg games). Moreover, theses games have potential applications for network deployment and routing. Since we aim to deal with continuous temporal and spatial spaces, we are faced with two difficulties. In our case, expressing the players' strategies as vector or a function would be difficult. In addition, the number of strategies is expected to be quiet large or infinite.

Search theory is another domain of game theory which deals with search efforts optimization. Most of the problems in this field are used to be modeled in discrete space and time, and therefore, operational research methods are often chosen to solve these problems. Eagle [8] and Washburn [22] were the first to study the problem of the optimization of spatio-temporal search efforts. Using a *branch and bound* solver, Eagle proposed to maximize a bound on the detection probability rather than the probability itself. Other approaches [1,3] only focus on spatial optimization without taking the target intelligence into account. Golen, meanwhile, recently [11] used a stochastic approach, more precisely, an evolutionary algorithm [2], but only optimize the sensors' discrete position.

Our approach, based on the splitting framework [6], is different because it allows us to optimize both space and time and we do not have to discretize these spaces anymore. In the sequel, we will try to point out that the temporal aspect is the most difficult part of our problem. This population-based optimization method can be linked to evolutionary algorithms.

In this article, we first present and model our problem. Then, we give the expression of the detection probability of interest and explain how we have applied the rare events simulation and the splitting algorithms to

Email addresses: mathieu.chouchane@lsis.org (Mathieu Chouchane), sebastien.paris@lsis.org (Sébastien Paris), legland@irisa.fr (François Le Gland), mustapha.ouladsine@lsis.org (Mustapha Ouladsine).

our problem, resulting in the generalized splitting for research efforts scheduling (GSRES) algorithm. Lastly, we illustrate our method with the *datum* problem and give prospects.

2 Problem presentation

In order to maximize the detection probability, until time T, of a smart and reactive moving target whose trajectory is unknown (and depends in practice on random variables), we have to deploy both spatially and temporally a limited number of active sensors in an area Ω (called the *operational theater*). Generally, Ω is a convex polygon with at least 3 vertices, but in most cases, it is a simple rectangle. As soon as a sensor is deployed, it is powered on and starts consuming energy ; it becomes out of service when its battery is empty. In this article, we don't study the sensors' power consumption, so therefore we only assume that a battery is powerful enough to make the sensor emit a maximum number of times during a fixed period. Because sensors are active, we assume they can only detect a target when they send a signal. Since they are not autonomous, they only ping when a moving commanding station requests it.

2.1 The target dynamic

We are only given an *a priori* on the starting point of the target trajectory. This *a priori* is weak if the starting point is randomly sampled in the search space Ω. On the contrary, a strong *a priori* means its initial position is sampled from a Gaussian pdf centered on the last seen position.

We then define a feasible trajectory $\boldsymbol{Y}_T \in \mathcal{Y}$ by a set of K points (or $K-1$ legs) composing it. So [17]

$$\begin{aligned} &\boldsymbol{Y}_T = \{\boldsymbol{y}_k\}_{k=0\dots K-1} \text{ where} \\ &\sum_{k=0}^{K-2} \frac{\|r_{k+1} - r_k\|_2}{v_k} + \frac{\|r_{K-1} - r_{K-2}\|_2}{v_{K-1}} = T, \end{aligned} \tag{1}$$

with v_k the target speed on the leg k. In this formula we note $\boldsymbol{y}_k \triangleq [r_{x,k}, v_{x,k}, r_{y,k}, v_{y,k}]_{k=0\dots K-1}$, where $r_{x,k}$ and $r_{y,k}$ define the target position and where $v_{x,k}$ and $v_{y,k}$ define the target velocity.

Moreover, if the target detects a signal of a sensor but is to far from it, it is able to avoid it before being detected and it is also able to memorize its position.

Basically, a target is instantly and surely detected if it enters the sensor's detection range. However, since the target is smart, if it comes close enough to a sensor which has just sent a signal, it detects it and learns all of its specifications. Thus, it may decide to avoid this threat or

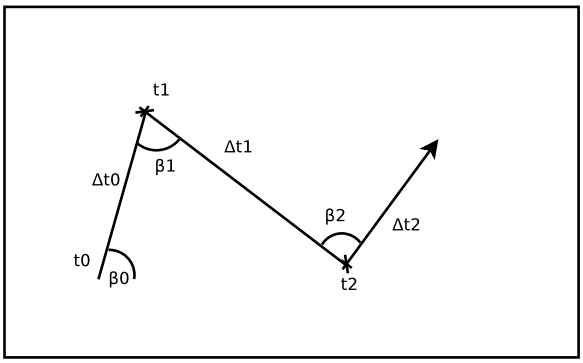


Fig. 1. Example of a leg-by-leg rectilinear trajectory (3 legs)

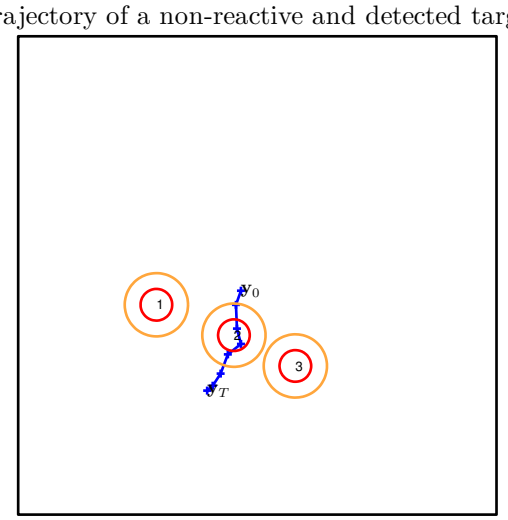


Fig. 2. The target starts from $\boldsymbol{y}_0$. After a short time, it is detected by sensor 2 and does not change its course. The trajectory ends at $\boldsymbol{y}_T$. In red, the detection circle. In orange, the counter-detection circle.

to come closer and start an avoidance later. In all cases, when the target is notified of the existence of a sensor, it changes its course before it enters the detection range. The target trajectory is then directly influenced by the partial knowledge of the target on the deployed search efforts $\mu_t(\boldsymbol{X})$.

For instance, $\mu_t(\boldsymbol{X}) = 0$ means that until time $t \leq T$, the target has not detected any search effort yet. On the contrary, if $\mu_t(\boldsymbol{X}) = \{1, 2\}$, the target has learned about a sensor existence and is trying to avoid it ($\mu_t(\boldsymbol{X}) = 1$) or to run away from it after a detection ($\mu_t(\boldsymbol{X}) = 2$). This instant of detection by a sensor $\boldsymbol{s}$ is denoted by $t_{\boldsymbol{s}}^{detect}$.

We also introduce the control vector $\boldsymbol{u}(\beta)$ which contains information about the change of course β. At last, we can give an expression of the motion state equation of

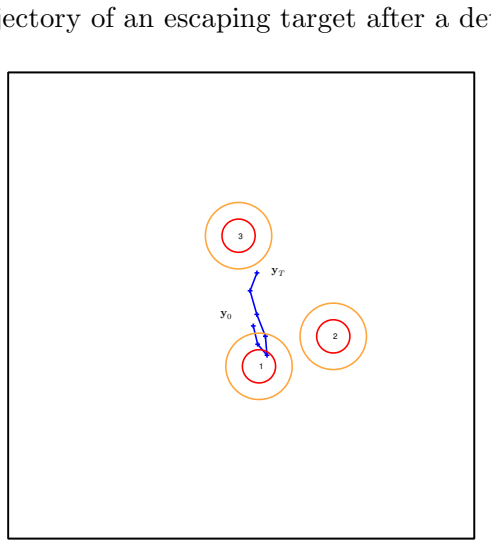


Fig. 3. The target starts from $\boldsymbol{y}_0$. After a while, it is detected by an active sensor (sensor 1) and immediately escapes by following a radial course.

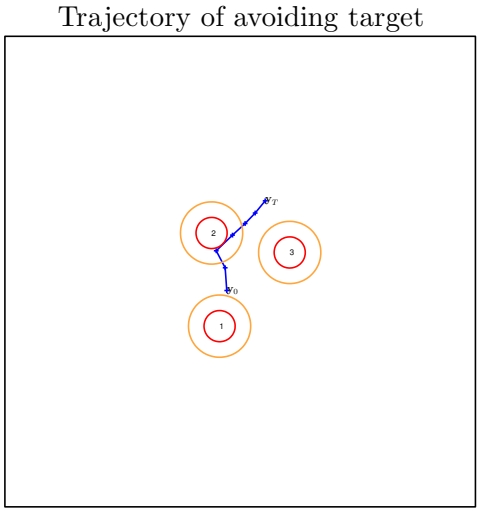


Fig. 4. The target starts from $\boldsymbol{y}_0$. After a while, it detects an active sensor (sensor 2) and avoids it.

our target:

$$\boldsymbol{y}_{k+1} = \boldsymbol{F}(\Delta t_k(\mu_k(\boldsymbol{X})))\boldsymbol{y}_k + \boldsymbol{u}_t(\beta_k(\mu_k(\boldsymbol{X}))) \ . \quad (2)$$

Here, Δt_k and β_k are two random variables and $\boldsymbol{F}(\Delta t_k(\mu_k(\boldsymbol{X})))$ is a state transition matrix associated with a rectilinear motion:

$$\boldsymbol{F}(\Delta t_k(\mu_k(\boldsymbol{X}))) = \begin{bmatrix} 1 & \Delta t_k(\mu_k(\boldsymbol{X})) & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \Delta t_k(\mu_k(\boldsymbol{X})) \\ 0 & 0 & 0 & 1 \end{bmatrix} . \quad (3)$$

More precisely, Δt_k is sampled from a truncated Gaussian law if $\mu_k(\boldsymbol{X}) = 0$ ($\Delta t_k \sim \mathcal{N}_t(\mu_{\Delta t}, \sigma^2_{\Delta t}, \mu_{\Delta t} - a, \mu_{\Delta t} + a), a \in \mathbb{R}$). When a detection occurs (*i.e.* $\mu_k(\boldsymbol{X}) = \{1, 2\}$), we end the current leg at t_s^{detect} and add an extra one, so that $K = K + 1$, $t_{k+2} = t_{k+1}$ and $t_{k+1} = t_s^{detect}$. Δt_{k+1} is then sampled from an uniform law as $\Delta t_{k+1} \sim \mathbb{1}([t_s^{detect}, t_{k+2}])$ (with $t_s^{detect} \sim \mathcal{U}([t_k, t_{k+2}])$). Whatever the value of $\mu_k(\boldsymbol{X})$, β_k is sampled from a truncated Gaussian law whose mean and bounds may vary depending on the last most threatening sensor met.

β_k, meanwhile, is sampled from a truncated Gaussian law ($\beta_k \sim \mathcal{N}_t(\mu_\beta, \sigma^2_\beta, \mu_\beta - a, \mu_\beta + a), a \in \mathbb{R}$) when $\mu_k(\boldsymbol{X}) = \{0, 1\}$, and from an uniform law when $\mu_k(\boldsymbol{X}) = 2$. More precisely, when $\mu_k(\boldsymbol{X}) = 0$, the mean of the Gaussian law μ_β is equal to the initial course of the trajectory β_0 or to its last course β_{k-1}, depending on the chosen dynamic. When $\mu_k(\boldsymbol{X}) = 1$, μ_β is directly defined by the last most threatening sensor met. Lastly, if the target is detected, *i.e.* $\mu_k(\boldsymbol{X}) = 2$, the new course of the target is defined by its previous course and the position of the detecting sensor.

2.2 The solution

Remember that we have a set of P sensors $\boldsymbol{s}_i$ that we may spatially and temporally deploy in our operational theatre $\Omega \times [0, T]$ in order to detect a smart and reactive target. A sensor is only operational for a limited amount of time (*e.g.* its battery life) and can therefore send a limited number of pings (E_{max} times maximum). Moreover, sensors are not autonomous and they are only able to send a signal if they are given the order by a commanding station. This is denoted by the visibility parameter until time $t \leq T$, $\varphi_t(\boldsymbol{X})$.

Before going further, we need to explain the following points:

- all the sensors must be in the search space Ω,
- sensors must have been set up before they can be activated,
- instants of activation must be in $[0; T]$.

Denote a solution by $\boldsymbol{X} \in \mathcal{X}$ (the set of feasible solutions), so we can define:

$$\boldsymbol{X} \triangleq \{\boldsymbol{s}_i(\boldsymbol{\tau}_i)\}_{i=1,\dots,P} \text{ with } \boldsymbol{\tau}_i = \{t_{i,1}, \dots, t_{i,j}\}_{j=1,\dots,np_i}. \quad (4)$$

$\boldsymbol{s}_i$ corresponds to the sensor i ($i \in \{1, \ldots, P\}$) position, while $\boldsymbol{\tau}_i$ is a vector containing its np_i instants of activation (in $[0, T]$). We also denote by $\mathcal{C}$ all the spatial and temporal constraints on the feasible solutions.

3 Evaluating the detection probability

We want to maximize the detection probability of a target until time T. This quantity is denoted by $S_T(\boldsymbol{X})$ and is given by the following equation:

$$S_T(\boldsymbol{X}) \triangleq \int_{Y_T \in \mathcal{Y}} f\left(\boldsymbol{Y}_T | \varphi_T(\boldsymbol{X}); \mathcal{C}\right) p\left(\boldsymbol{Y}_T | \mu_T(\boldsymbol{X}); \mathcal{C}\right) \, d\boldsymbol{Y}_T \ . \tag{5}$$

Here, $f\left(\boldsymbol{Y}_T | \varphi_T(\boldsymbol{X}); \mathcal{C}\right)$ is a cookie-cutter cost function that takes the value 1 if the studied trajectory $\boldsymbol{Y}_T$ satisfies some defined criteria (such as a number of detections and a number of avoidances) and 0 otherwise, *i.e.* $f\left(\boldsymbol{Y}_T | \varphi_T(\boldsymbol{X}); \mathcal{C}\right) = \mathbb{1}_{\{\boldsymbol{Y}_T \in A(\boldsymbol{X}, \mathcal{C})\}}$ where $A(\boldsymbol{X}, \mathcal{C})$ is the set of trajectories which are detected by the solution $\boldsymbol{X}$ and which respect the constraints $\mathcal{C}$. It depends on the visibility of the solution $\varphi_T(\boldsymbol{X})$. $p\left(\boldsymbol{Y}_T | \mu_T(\boldsymbol{X}); \mathcal{C}\right)$ is the conditional pdf used to generate the target trajectories and depends on the target intelligence $\mu_T(\boldsymbol{X})$. As this is not the goal of this article, we do not give any more information on how we have implemented this cost function. Unfortunately, $S_T(\boldsymbol{X})$ is an integral with respect to the probability distribution of the (random, solution-dependant) target trajectory $\boldsymbol{Y}$ and its analytical expression is not available. A first approach should be to use the drude Monte Carlo method to obtain an unbiased estimator of $S_T(\boldsymbol{X})$, $\widehat{S}_T(\boldsymbol{X})$:

$$\begin{array}{c} \widehat{S}_T(\boldsymbol{X}) = \frac{1}{N} \sum_{i=1}^{N} f(\boldsymbol{Y}_T^i | \varphi_T(\boldsymbol{X}); \mathcal{C}), \text{ where} \\ \boldsymbol{Y}_T^i \sim p(\boldsymbol{Y}_T | \mu_T(\boldsymbol{X}); \mathcal{C}). \end{array} \tag{6}$$

The trajectories $\boldsymbol{Y}_T^i$ are recursively generated using the motion state equation (2). To be concrete, we generate a large number of feasible trajectories $\boldsymbol{Y}_T^i, i = 1, \ldots, N$ and evaluate $f\left(\boldsymbol{Y}_T^i | \varphi_T(\boldsymbol{X}); \mathcal{C}\right)$.

Note that the relative error associated with $\widehat{S}_T(\boldsymbol{X})$ given by the CMC estimator is

$$RE_{CMC}(\widehat{S}_T(\boldsymbol{X})) = \frac{\sqrt{1 - S_T(\boldsymbol{X})}}{\sqrt{N S_T(\boldsymbol{X})}} \ . \tag{7}$$

Also remark that the smaller the probability to estimate is, the larger the relative error is. To reduce this error, we have to increase the number of trajectories N. Knowing the probability we are meeting are above 10^{-3} (if they were below, planning would be useless), we have chosen $N \geq 50000$.

The optimal solution we search, denoted by $\boldsymbol{X}^\star$ is defined by this formula:

$$\boldsymbol{X}^\star \triangleq \arg\max_{\boldsymbol{X} \in \mathcal{X}} \{S_T(\boldsymbol{X})\}, \ \boldsymbol{X} \in \mathcal{X}. \tag{8}$$

We also define $\gamma^\star \triangleq S_T(\boldsymbol{X}^\star)$ as the maximum detection probability we can reach. Let $\boldsymbol{X}^\dagger \triangleq \arg\max\{\widehat{S}_T(\boldsymbol{X})\}$ be the solution obtained when maximizing the approached probability $\gamma^\dagger = \widehat{S}_T(\boldsymbol{X}^\dagger)$. For the sequel we focus on computing $\gamma^\dagger$ and $\boldsymbol{X}^\dagger$. Indeed, if N is large enough, we can expect that these are good approximations of $\gamma^\star$ and $\boldsymbol{X}^\star$ respectively. A sufficient condition for this to hold is that $\widehat{S}_T(\boldsymbol{X})$ converges to $S_T(\boldsymbol{X})$ when $N \to \infty$ uniformly in $\boldsymbol{X} \in \mathcal{X}$.

Splitting theory is based on the observation that maximizing $S_T(\boldsymbol{X})$ (in practice $\widehat{S}_T(\boldsymbol{X})$) is similar

- to estimating the probability

$$\ell(\gamma) = \mathbb{P}(\widehat{S}_T(\boldsymbol{X}) \geq \gamma) = \int_{\mathcal{X}} \mathbb{1}_{\{\widehat{S}_T(\boldsymbol{X}) \geq \gamma\}} q(\boldsymbol{X}, \mathcal{C}) d\boldsymbol{X}, \tag{9}$$

  where $q(\boldsymbol{X}, \mathcal{C})$ is some positive probability density on $\mathcal{X}$, with the idea that this probability decreases and reaches zero when γ increases toward the (unknown) value $\gamma^\dagger$ and is identically zero beyond this value, hence the characterization

$$\gamma^\dagger = \inf\{\gamma \geq 0 : \ell(\gamma) = 0\} \ , \tag{10}$$

- and simultaneously to sampling the set

$$\mathcal{X}_\gamma = \{\boldsymbol{X} \in \mathcal{X} : \widehat{S}_T(\boldsymbol{X}) \geq \gamma\} \subset \mathcal{X} \ , \tag{11}$$

  with the idea that this set decreases toward $\boldsymbol{X}^\dagger$ when γ increases toward the (unknown) value $\gamma^\dagger$ and reduces to 0 beyond this value.

However, when γ goes to $\gamma^\dagger$, the event $\{\widehat{S}_T(\boldsymbol{X}) \geq \gamma\}$ becomes rarer and rarer, and consequently the CMC estimator

$$\ell(\gamma) = \frac{1}{C} \sum_{i=1}^{C} \mathbb{1}_{\{\widehat{S}_T(\boldsymbol{X}_i) \geq \gamma\}} \text{ where } \boldsymbol{X}_i \sim q(\boldsymbol{X}, \mathcal{C}) \tag{12}$$

has a relative error

$$RE_{CMC}(\ell(\gamma)) = \frac{\sqrt{1 - \ell(\gamma)}}{\sqrt{C \ell(\gamma)}} \tag{13}$$

that increases to infinity. In order to reduce this relative error, we should increase the sample size C, but then, we would be faced with a computational explosion. Moreover, when γ goes to $\gamma^\dagger$, it becomes harder and harder to produce samples from $q(\boldsymbol{X}, \mathcal{C})$ that would be close to $\boldsymbol{X}^\dagger$.

To address this problem, a technique called generalized splitting [5] and derived from Diaconis, Holmes and Ross researches [9] on MCMC (Markov chain Monte Carlo) allows us to compute $\ell(\gamma)$ in an easier and more precise way. For an optimization problem, we will find at least one solution that maximizes our criteria among all the solutions sampled to compute our probability of interest.

If we define a sequence of increasing thresholds γ, such that $\gamma_0 \geq \gamma_1 \geq \ldots \geq \gamma_L$ (with $\gamma_L \leq \gamma^\star$), we can rewrite $\ell(\gamma)$ as the following product of conditional probabilities:

$$\begin{aligned} \ell(\gamma) &= \mathbb{P}_q\left(\widehat{S}_T(\boldsymbol{X}) \geq \gamma_0\right) \prod_{l=1}^{L} \mathbb{P}_q\left(\widehat{S}_T(\boldsymbol{X}) \geq \gamma_l | \widehat{S}_T(\boldsymbol{X}) \geq \gamma_{l-1}\right) \\ &= c_0 \prod_{l=1}^{L} c_l \ . \end{aligned} \tag{14}$$

where

$$\begin{aligned} c_l &= \mathbb{P}_q\left(\widehat{S}_T(\boldsymbol{X}) \geq \gamma_l | \widehat{S}_T(\boldsymbol{X}) \geq \gamma_{l-1}\right) \\ &= \frac{\mathbb{P}_q\left(\widehat{S}_T(\boldsymbol{X}) \geq \gamma_l, \widehat{S}_T(\boldsymbol{X}) \geq \gamma_{l-1}\right)}{\ell(\gamma_{l-1})} \\ &= \int_{\boldsymbol{X} \in \mathcal{X}} \mathbb{1}_{\{\widehat{S}_T(\boldsymbol{X}) \geq \gamma_{l-1}\}} \frac{\mathbb{1}_{\{\widehat{S}_T(\boldsymbol{X}) \geq \gamma_l\}} q(\boldsymbol{X}, \mathcal{C})}{\ell(\gamma_{l-1})} \, d\boldsymbol{X} \\ &= \int_{\boldsymbol{X} \in \mathcal{X}} \mathbb{1}_{\{\widehat{S}_T(\boldsymbol{X}) \geq \gamma_l\}} g^\star_{l-1}(X; \gamma_{l-1}, \mathcal{C}) \, d\boldsymbol{X} \\ &= \mathbb{P}_{g^\star_{l-1}}(\widehat{S}_T(\boldsymbol{X}) \geq \gamma_l) \ . \end{aligned} \tag{15}$$

and where the importance sampling density [18] $g^\star_{l-1}(X; \gamma_{l-1}, \mathcal{C}) = \frac{\mathbb{1}_{\{\widehat{S}_T(\boldsymbol{X}) \geq \gamma_{l-1}\}} q(\boldsymbol{X}, \mathcal{C})}{\ell(\gamma_{l-1})}$ is precisely the conditional density of $\boldsymbol{X}$, given that $\widehat{S}_T(\boldsymbol{X}) \geq \gamma_{l-1}$. Remark that the support of this density $g^\star_{l-1}(\boldsymbol{X}; \gamma_{l-1}, \mathcal{C})$ is precisely the set $\{\boldsymbol{X} \in \mathcal{X} : \widehat{S}_T(\boldsymbol{X}) \geq \gamma_{l-1}\}$.

If we know how to draw independent and identically distributed random variables $\boldsymbol{X}_i$ over $\mathcal{X}_{l-1} \in \mathcal{X}$ from this importance sampling function, $\ell(\gamma)$ can be rewritten:

$$\begin{aligned} \ell(\gamma) &= \mathbb{P}_q\left(\widehat{S}_T(\boldsymbol{X}) \geq \gamma_0\right) \prod_{l=1}^{L} \mathbb{P}_{g^\star_{l-1}}\left(\widehat{S}_T(\boldsymbol{X}) \geq \gamma_l\right) \\ &= c_0 \prod_{l=1}^{L} c_l \ . \end{aligned} \tag{16}$$

With a judicious choice or a fair estimation of the $\{\gamma_l\}$ sequence, the event $\{S_T(\boldsymbol{X}) \geq \gamma_l\}$ is no longer a rare event (generally, $c_l \in \left[10^{-3}, 10^{-2}\right]$) under the distribution $g^\star_{l-1}(\boldsymbol{X}, \gamma_{l-1}, \mathcal{C})$ and therefore the c_l quantities can now be well approximated through a CMC estimator. Hence, a CMC estimator of $\ell(\gamma)$ is:

$$\widehat{\ell}(\gamma) = \prod_{l=0}^{L} \widehat{c}_l, \tag{17}$$

where $\widehat{c}_l = \frac{1}{C} \sum_{i=1}^{C} \mathbb{1}_{\{\widehat{S}_T(\boldsymbol{X}_i) \geq \gamma_l\}}$ and where $\boldsymbol{X}_i \sim g^\star_{l-1}(\boldsymbol{X}; \gamma_{l-1}, \mathcal{C})$.

4 The splitting algorithm

We will now describe the general splitting algorithm applied to our optimization problem.

Generate a population $\mathcal{X}_0 = \{\boldsymbol{X}_1, \ldots, \boldsymbol{X}_C\} \sim q(\boldsymbol{X}; \mathcal{C})$ of C feasible solutions (that respect all the constraints in $\mathcal{C}$) and initialize an iteration counter $l = 1$. Evaluate the scores of the solutions $\mathcal{S}_0 = \{\widehat{S}_T(\boldsymbol{X}_i)\}$ and sort $\mathcal{S}_0$ in decreasing order such that $\widehat{S}_T(\boldsymbol{X}_{j(1)}) \geq \widehat{S}_T(\boldsymbol{X}_{j(2)}) \geq \ldots \geq \widehat{S}_T(\boldsymbol{X}_{j(C)})$. We obtain $\widehat{\gamma}_0 = \widehat{S}_T(\boldsymbol{X}_{j(C_0)})$ with $C_0 = \lfloor \rho C \rfloor$. Set $\widetilde{\mathcal{X}}_0 = \{\widetilde{\boldsymbol{X}}_i\}_{i=1,\ldots,C_0} \subset \mathcal{X}_0$ such that $\widehat{S}_T(\widetilde{\boldsymbol{X}}_i) \geq \gamma_0$. Remark that $\widetilde{\boldsymbol{X}}_i \sim g^\star_0(\boldsymbol{X}; \gamma_0, \mathcal{C})$ for $i = 1, \ldots, C_0$.

At iteration l of the algorithm, $\widetilde{\mathcal{X}}_{l-1}$ contains only $\rho\%$ samples of the initial population $\mathcal{X}_{l-1}$ whose score is above the current threshold $\widehat{\gamma}_{l-1}$. Since we want to work with a constant number C of solutions, we have to complete this set. This is done in two steps. First, we use the bootstrap method or the ADAM cloning method to repopulate our set of solutions. The bootstrap method consists of sampling uniformly with replacement C times from the population $\widetilde{\mathcal{X}}_{l-1}$. ADAM cloning method, meanwhile, consist of making $\left\lfloor \frac{C}{C_{l-1}} \right\rfloor + B_i (i = 1, \ldots, C_{l-1})$ copies of each sample. Here each $B_1, \ldots, B_{l-1}$ are $Ber(1/2)$ random variables conditional on $\sum_{i=1}^{C_{l-1}} B_i = C \mod C_{l-1}$ and $[B_1, \ldots, B_{C_{l-1}}]$ is a binary vector with joint pdf $\mathbb{P}(B_1 = b_1, \ldots, B_{C_{l-1}} = b_{C_{l-1}}) = \frac{(C_{l-1}-r)!r!}{C_{l-1}!} \mathbb{1}_{\{b_1 + \ldots + b_{C_{l-1}} = r\}}$, $b_i \in \{0, 1\}$, where $r = C \mod C_{l-1}$. Unfortunately, the samples of the

completed set denoted by $\widetilde{\mathcal{X}}_{l-1}^{boot/clon}$ are identically distributed but not independent.

To address this problem, we apply a *random* Gibbs sampler $\pi_{l-1}(\boldsymbol{X}|\widetilde{\boldsymbol{X}}_{l-1};\mathcal{C}) = \frac{1}{C_{l-1}}\sum_{i=1}^{C_{l-1}} \kappa_{l-1}(\boldsymbol{X}|\widetilde{\boldsymbol{X}}_i;\mathcal{C})$ to each sample $\widetilde{X}_i$ of $\mathcal{X}_{l-1}^{boot/clon}$ to obtain $\mathcal{X}_l = \{\boldsymbol{X}_i\}$ such that $\boldsymbol{X}_i \sim g_{l-1}^{\star}(\boldsymbol{X};\widehat{\gamma}_{l-1},\mathcal{C})$ for $i = 1,\ldots,C$. Here, κ_{l-1} is the transition density of a Markov chain starting from $\widetilde{\mathcal{X}}_{l-1}^{boot/clon}$ and with stationary pdf $g_{l-1}^{\star}$. For our problem, we have the transition density κ_{l-1} defined by:

$$\kappa_{l-1}\left(\boldsymbol{X}|\widetilde{\boldsymbol{X}}_i\right) = \sum_{j=1}^{6}\lambda_j \prod_{r=1}^{b_l} m_j\left(\boldsymbol{X}_i^r|\widetilde{\boldsymbol{X}}_i^{-r}\right) \ , \qquad (18)$$

where $\boldsymbol{X}_i^r$ denotes the component r of a solution and $\boldsymbol{X}_i^{-r}$, all the components of $\widetilde{\boldsymbol{X}}_i$ but the r one. The λ_j are the probabilities of updating one component at a time, given that $\sum_j \lambda_j = 1$ and the m_j are the conditional pdf associated to the 6 moves (described below). We could also use a *systematic* Gibbs sampler, whose transition density is defined by

$$\kappa_{l-1}\left(\boldsymbol{X}|\widetilde{\boldsymbol{X}}_i\right) = \sum_{j=1}^{6}\lambda_j \prod_{r=1}^{P_{max}} m_j\left(\boldsymbol{X}_i^r|\widetilde{\boldsymbol{X}}_i^{-r}\right) \ . \qquad (19)$$

Notice that we have chosen a random Gibbs sampler with b_l random updates of components of a solution $\widetilde{X}_i$, while with a systematic Gibbs sampler, we would have updated all the components of $\widetilde{X}_i$ in a fixed order. The number of random updates b_l varies during the simulation in this way: $b_l = b_0 + \alpha l$ where $\alpha \in \mathbb{R}_+^{\star}$. For the first iterations, $b_l < P$ and therefore this approach is faster than a systematic Gibbs sampler. On the contrary, when l is close to L, $b_l \geq P$. Thus, we do more updates than a systematic Gibbs sampler would do but we maintain more diversity in our solutions.

Since we do not know how to update a solution in a way it still satisfies the constraints $\mathcal{C}$, we first recursively propagate the modifications starting from the sensor/activation we have modified in the sequence of activations. Then we check its feasibility, that is, if it respects all the spatial and temporal constraints $\mathcal{C}$. We apply acceptance–rejection (for a limited times) to each updated component until we find a feasible solution. Considering that the cost function S_T also verifies the consistency of a solution, an updated solution $\boldsymbol{X}_i$ from $\widetilde{\mathcal{X}}_{i,l-1}^{boot/clon}$ is then accepted with probability $\mathbb{1}_{\{S_T(\boldsymbol{X}_i)\geq\widehat{\gamma}_{l-1}\}}$. Henceforth, all the $\boldsymbol{X}_i$ should be iid.

In the same way as before, evaluate the scores $\mathcal{S}_l = \{\widehat{S}_T(\boldsymbol{X}_i)\}$, $\boldsymbol{X}_i \in \mathcal{X}_l$ and sort this set in decreasing order such that $\widehat{S}_T(\boldsymbol{X}_{j(1)}) \geq \widehat{S}_T(\boldsymbol{X}_{j(2)}) \geq \ldots \geq \widehat{S}_T(\boldsymbol{X}_{j(C)})$. We obtain $\widehat{\gamma}_l = \widehat{S}_T(\boldsymbol{X}_{j(C_l)})$ with $C_l = \lfloor \rho C \rfloor$. Deduct that $\widetilde{\boldsymbol{X}}_l = \boldsymbol{X}_{j(1)} = \arg\max\{\widehat{S}_T(\boldsymbol{X}_i)\}$, $i = 1,\ldots,C$ and $\widetilde{\gamma}_l = \widehat{S}_T(\widetilde{\boldsymbol{X}}_l)$ respectively the best solution at iteration and its score at iteration l. In addition, denote by $\widehat{\gamma}_{0:l} = \max\{\widetilde{\gamma}_l, \widehat{\gamma}_{0:l-1}\}$ and $\widehat{\boldsymbol{X}}_{0:l} = \widetilde{\boldsymbol{X}}_l$ if $\widetilde{\gamma}_l > \widehat{\gamma}_{0:l-1}$ or $\widehat{\boldsymbol{X}}_{0:l} = \widehat{\boldsymbol{X}}_{0:l-1}$ otherwise, respectively the best detection probability and its associated solution ever met until iteration l.

If one of the stopping criteria is satisfied, stop the algorithm and give $\widehat{\boldsymbol{X}}_{0:l}$ as an estimator of the optimal solution.

In practice, the threshold $\widehat{\gamma}_l$ is the ρ quantile of the sample $\mathcal{X}_l$.

4.1 The Gibbs sampler moves

Before we go further, let us introduce and recall a few notations. $\boldsymbol{s}_i \triangleq [s_{i_x}; s_{i_y}]$ denotes the ith sensor position (and more generally the ith sensor), P is the number of sensors in the current solution, P_{max} is the maximum number of sensors, np_i stands for the activations' number for sensor i while $t_{i,\{1,\ldots,np_i\}}$ and $\boldsymbol{\tau}_i$ respectively are the instants of activation of sensor i and the set of activation times associated with sensor i. Also denote by $t_{\boldsymbol{s}_i}$ the set up duration of the sensor $\boldsymbol{s}_i$. Remark that a sensor whose instants of activation are negative is considered as disabled. Consequently, deleting an instant of activation consists of assigning a negative value to this instant. Removing a sensor is then equivalent to deleting all of its instants of activation and ignoring it. Now we give more details on the 6 moves of our Gibbs sampler and the associated conditional pdf.

(1) **Add a sensor**. The conditional pdf m_1 can be defined in two steps. Sample a position $\boldsymbol{s}'_{P+1}$ from $\mathcal{U}(\Omega;\mathcal{C})$ for the new sensor. Then draw its first instant of activation $t'_{P+1,1} \sim \mathcal{U}([t_{\boldsymbol{s}_{P+1}}, T])$. m_1 is proportional to (up to a normalization constraint):

$$\begin{aligned} &m_1\left(\boldsymbol{X}_i^{P+1}|\widetilde{\boldsymbol{X}}_i^{-(P+1)}\right) \propto \\ &\mathcal{U}(\Omega;\mathcal{C})\,\mathcal{U}([t_{\boldsymbol{s}_{P+1}}, T])\ \mathbb{1}_{\{S_T(\boldsymbol{X}_i)\geq\gamma_{l-1}\}} \ . \end{aligned} \qquad (20)$$

(2) **Add one instant of activation**. First, choose a sensor randomly *i.e.* draw j uniformly in $\{1,\ldots,P\}$. If $np_j < np_{max}$ then draw $t'_{j,np_j+1} \sim \mathcal{U}([t_{j,1}, T])$. m_2 is proportional to:

$$\begin{aligned} &m_2\left(\boldsymbol{X}_i^{j}|\widetilde{\boldsymbol{X}}_i^{-j}\right) \propto \\ &\mathcal{U}(\{1,\ldots,P\})\,\mathcal{U}([t_{j,1}, T])\ \mathbb{1}_{\{S_T(\boldsymbol{X}_i)\geq\gamma_{l-1}\}} \ . \end{aligned} \qquad (21)$$

(3) **Remove a sensor**. To apply the move m_3, we apply the following steps : choose a sensor randomly,

i.e. draw j uniformly in $\{1,\ldots,P\}$. Then delete all of its instants of activation and mark it as disabled. So we have

$$m_3\left(\boldsymbol{X}_i^j|\widetilde{\boldsymbol{X}}_i^{-j}\right) \propto \mathcal{U}(\{1,\ldots,P\})\ \mathbb{1}_{\{S_T(\boldsymbol{X}_i)\geq\gamma_{l-1}\}}\ . \tag{22}$$

(4) **Remove an instant of activation**. Choose a sensor randomly *i.e.* draw j uniformly in $\{1,\ldots,P\}$. We assume that $np_j > 1$. Choose an instant of activation $t_{j,k}$, *i.e.*, draw k uniformly in $\{2,\ldots,np_j\}$. Delete $t'_{j,k}$. The conditional pdf m_4 is defined as below:

$$m_4\left(\boldsymbol{X}_i^j|\widetilde{\boldsymbol{X}}_i^{-j}\right) \propto \mathcal{U}(\{1,\ldots,P\})\ \mathcal{U}(\{1,\ldots,np_j\})\ \mathbb{1}_{\{S_T(\boldsymbol{X}_i)\geq\gamma_{l-1}\}}\ . \tag{23}$$

(5) **Move a sensor**. Select a sensor $\boldsymbol{s}_j$ randomly, *i.e.* draw j uniformly in $\{1,\ldots,P\}$. Then, draw $\boldsymbol{s}'_j \sim \sum_{k=1}^2 w_k\ \mathcal{N}(\boldsymbol{s}_j,\boldsymbol{\Sigma}_k^2)$ with $\sum_{k=1}^2 w_k = 1$. Notice that the weights w_k may evolve during the optimization in order to promote one of the move versus the other. For this mixture of two Gaussian pdf the covariance of the first Gaussian defines a small move while the covariance of the second Gaussian defines a larger move. Here is the formula of this conditional pdf:

$$m_5\left(\boldsymbol{X}_i^j|\widetilde{\boldsymbol{X}}_i^{-j}\right) \propto \mathcal{U}(\{1,\ldots,P\})\ \sum_{k=1}^2 w_k\ \mathcal{N}(\boldsymbol{s}_j,\boldsymbol{\Sigma}_k^2)\ \mathbb{1}_{\{S_T(\boldsymbol{X}_i)\geq\gamma_{l-1}\}}\ . \tag{24}$$

(6) **Swap two sensors**. If we assume there are at least 2 active sensors, select two sensors $\boldsymbol{s}_k$ and $\boldsymbol{s}_r$ with k uniformly drawn in $\{1,\ldots,P\}$ and r uniformly drawn in $\{1,\ldots,P\}\setminus\{k\}$. For all $k = 2,\ldots,np_k$, delete $t'_{k,j}$ and $t'_{r,j}$ for all $r = 2,\ldots,np_r$. Swap their first instant of activation: $t'_{k,1}$ and $t'_{r,1}$. The conditional bivariate pdf m_6 is then defined by:

$$m_6\left(\boldsymbol{X}_i^k,\boldsymbol{X}_i^r|\widetilde{\boldsymbol{X}}_i^{-k,-r}\right) \propto \mathcal{U}(\{1,\ldots,P\})\ \mathcal{U}(\{1,\ldots,P\}\setminus\{k\})\ \mathbb{1}_{\{S_T(\boldsymbol{X}_i)\geq\gamma_{l-1}\}}\ , \tag{25}$$

where $(\boldsymbol{X}_i^k,\boldsymbol{X}_i^r) \in \{(\widetilde{\boldsymbol{X}}_i^k,\widetilde{\boldsymbol{X}}_i^r),(\widetilde{\boldsymbol{X}}_i^r,\widetilde{\boldsymbol{X}}_i^k)\}$.

4.2 The GSRES algorithm

Algorithm 1 (GSRES) *Given parameter ρ, sample number C and number of burn-in iterations b_l of the Gibbs sampler, follow the forthcoming steps:*

***1. Initialization**. Set a counter $l = 1$. Generate C feasible solutions $\{\boldsymbol{X}_i\}, i = 1,\ldots,C$ and denote $\mathcal{X}_0$ the set containing them. Note that $\boldsymbol{X}_i \sim q(\boldsymbol{X};\mathcal{C})$. Evaluate scores $\mathcal{S}_0 = \{\widehat{S}_T(\boldsymbol{X}_i)\}$ and sort in decreasing order $\mathcal{S}_0$ such that $\widehat{S}_T(\boldsymbol{X}_{j(1)}) \geq \widehat{S}_T(\boldsymbol{X}_{j(2)}) \geq \ldots \geq \widehat{S}_T(\boldsymbol{X}_{j(C)})$. We obtain $\widehat{\gamma}_0 = \widehat{S}_T(\boldsymbol{X}_{j(C_0)})$ with $C_0 = \lfloor \rho C \rfloor$. Define $\widetilde{\boldsymbol{X}}_0 = \widehat{\boldsymbol{X}}_{0:0} = \boldsymbol{X}_{j(1)}$, $\widetilde{\gamma}_l = \widehat{\gamma}_{0:0} = \widehat{S}_T(\boldsymbol{X}_{j(1)})$.*

***2. Selection**. Let $\widetilde{\mathcal{X}}_{l-1} = \{\widetilde{\boldsymbol{X}}_1,...,\widetilde{\boldsymbol{X}}_{C_{l-1}}\}$ be the subset of the population $\{\boldsymbol{X}_1,...,\boldsymbol{X}_C\}$ for which $\widehat{S}_T(\widetilde{\boldsymbol{X}}_i) \geq \widehat{\gamma}_{l-1}$. $\widetilde{\mathcal{X}}_{l-1}$ contains ρ% of the population. Notice that $\widetilde{\boldsymbol{X}}_i \sim g^\star_{l-1}(\boldsymbol{X};\widehat{\gamma}_{l-1},\mathcal{C})$ for $i = 1,\ldots,C_{l-1}$.*

***3. Repopulation**. Apply one of these methods:*

- *Bootstraping: sample uniformly with replacement C times from the population $\widetilde{\mathcal{X}}_{l-1}$ to define the temporary set of C solutions $\mathcal{X}_{l-1}^{boot}$.*
- *ADAM Cloning: make $\left\lfloor \frac{C}{C_l} \right\rfloor + B_i (i = 1,\ldots,C_l)$ copies of each population sample $\widetilde{\mathcal{X}}_{l-1}$. Here each $B_1,\ldots,B_{C_l}$ are $Ber(1/2)$ random variables conditional on $\sum_{i=1}^{C_l} B_i = C \mod C_l$. We then define the temporary set of C solutions $\mathcal{X}_{l-1}^{clon}$.*

***4. Gibbs sampler**. Apply a random Gibbs sampler $\pi_{l-1}(\boldsymbol{X}|\widetilde{\boldsymbol{X}}_{l-1};\mathcal{C}) = \frac{1}{C_{l-1}}\sum_{i=1}^{C_{l-1}} \kappa_{l-1}(\boldsymbol{X}|\widetilde{\boldsymbol{X}}_i;\mathcal{C})$ with b_l burn-in iterations and the transition density κ_{l-1} to each sample of $\mathcal{X}_{l-1}^{boot/clon}$ (see section 4.1) to obtain $\mathcal{X}_l = \{\boldsymbol{X}_i\}$ such that $\boldsymbol{X}_i \sim g^\star_{l-1}(\boldsymbol{X};\widehat{\gamma}_{l-1},\mathcal{C})$ for $i = 1,\ldots,C$. Notice that the $\boldsymbol{X}_i, i = 1,\ldots,C$ should be approximately independent and identically distributed.*

***5. Estimation**. Evaluate scores $\mathcal{S}_l = \{\widehat{S}_T(\boldsymbol{X}_i)\}$, $\boldsymbol{X}_i \in \mathcal{X}_l$. Sort in decreasing order $\mathcal{S}_l$ such that $\widehat{S}_T(\boldsymbol{X}_{j(1)}) \geq \widehat{S}_T(\boldsymbol{X}_{j(2)}) \geq \ldots \geq \widehat{S}_T(\boldsymbol{X}_{j(C)})$. We obtain $\widehat{\gamma}_l = \widehat{S}_T(\boldsymbol{X}_{j(C_l)})$ with $C_l = \lfloor \rho C \rfloor$. Deduct that $\widetilde{\boldsymbol{X}}_l = \boldsymbol{X}_{j(1)}$, $\widetilde{\gamma}_l = \widehat{S}_T(\boldsymbol{X}_{j(1)})$, $\widehat{\boldsymbol{X}}_{0:l} = \widetilde{\boldsymbol{X}}_l$ if $\widetilde{\gamma}_l > \widehat{\gamma}_{0:l-1}$, else $\widehat{\boldsymbol{X}}_{0:l} = \widehat{\boldsymbol{X}}_{0:l-1}$ and $\widehat{\gamma}_{0:l} = \max\{\widetilde{\gamma}_l, \widehat{\gamma}_{0:l-1}\}$.*

***6. Stopping condition**. If one of the stopping condition is reached, stop the algorithm and give $\widehat{\boldsymbol{X}}_{0:l}$ as an estimator of the optimal solution. Else $l = l + 1$ and go back to step 2.*

For our problem, we implement a customized version of the splitting method. To begin the computation, we generate an initial pool of feasible solutions with $q(.;\mathcal{C})$. Since a solution and the carrier trajectory are closely linked, we use our trajectory generator to obtain a pool of initial solutions that respect the whole constraints set $\mathcal{C}$.

Lastly, to ensure our algorithm will not converge and stay into a local extremum, we developed a simple heuristic. If the current maximum score and the current threshold do not increase for a chosen number of times, we automatically reduce the value of the threshold. Through

the decrease of the threshold, we start again to accept the feasible solutions generated by the moves and therefore, we reintroduce some diversity in the pool of solutions.

5 Illustrative example

The first result we present here concerns a scenario in which a target is running away from the position it has just been detected. Its initial position is drawn from a Gaussian law centered on $\Omega/2$ and with a variance σ^2_{target}. Moreover, the target is supposed to be smart and reactive and therefore, while it is running away, it tries to avoid being detected another time. Considering that the research starts with a delay of t_c^{aoz} which represent the time of arrival of the hunter, we aim to maximize the chances to detect the target during the time T. We use $P_{max} = 10$ sensors that are able to ping only once. For this simulation, we use $C = 800$ solutions, $N = 70000$ trajectories, $b_0 = 2$, $b_l = b_0 + 0.2\ l$ and decide to keep 10% of elites ($\rho = 0.1$). We also let the algorithm perform up to 50 iterations.

Because our algorithm is not able yet to adjust the number of sensors considering the cost of their deployment, we have chosen to work with a constant number of sensors. However, we have allowed the removal of a sensor if it is directly followed by an addition of a new sensor. We have used two of the six moves we have defined above : move a sensor and a combination of removing a sensor followed by the addition of a new sensor. The probability of both moves to occur is 0.5.

In the best solution we obtain, the sensors position and activation describe a spiral. This result, illustrated in the figure 5, is related to the studies of Washburn [21] and Son [19] for an only-spatial optimization case, *i.e.*, when the target is not able to avoid the sensor ("myope" case). In this context, the best spatial sensor deployment designs an Archimedean spiral.

In the sequel, we plot the optimization evolution behaviour versus iterations $\widehat{\gamma}_{0:l} = \widehat{S}_T(\widehat{\mathbf{X}}_{0:l})$ and $\mathbb{E}[\widehat{S}_T(\widehat{\mathbf{X}}_l)]$ which represents the mean score of the current population (figure 6). Both are smoothly increasing in a logarithmic way. On the figure 7 we see that the support of scores pdf, much large at the beginning ($l = 0$), becomes thinner and thinner and converges toward a Dirac pdf as the optimization is conducted. Moreover, $\widehat{\gamma}_{0:l}$ increases and the standard deviation of the distribution decreases. Once the optimization is over, we observe that the detection probability reaches 0.9406 whereas when $l = 0$, the best score is below 0.25 and the mean scores $\mathbb{E}\left[\widehat{S}_T(\boldsymbol{X})\right]$ is equal to 0.0187 with a large standard deviation. This gap illustrates the efficiency of our approach.

Figure 8, visualizes the result of two other simulations for wich the planification is not as optimal as for the previous example. Nevertheless, the detection probabilities associated with these deployments are above 0.93.

To compare our best solution (see figure 5) with Son's solution, we have computed the best track spacing of the Archimedean spiral (denoted by TS) for our sce-

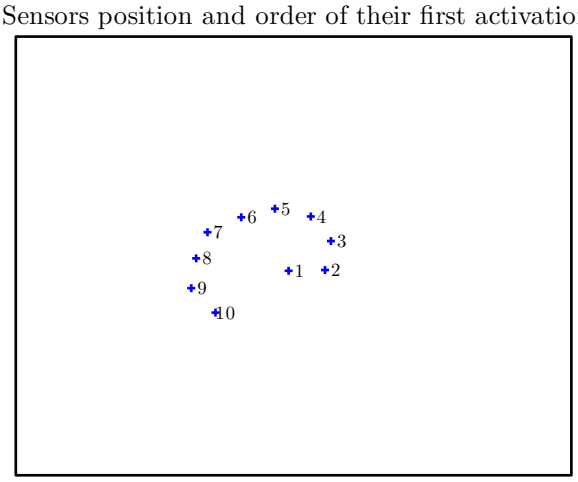


Fig. 5. Graphic of $\boldsymbol{X}^\dagger$: position and activation order of the 10 sensors.

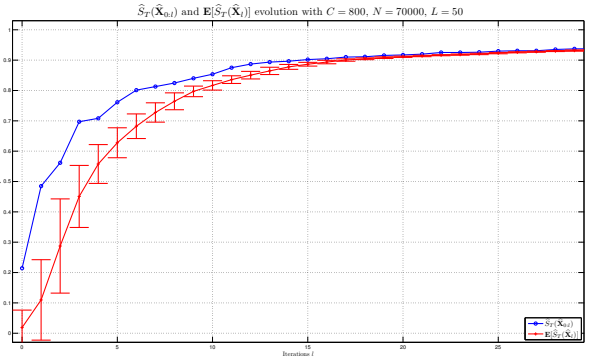


Fig. 6. In blue $\widehat{S}_T(\widehat{\mathbf{X}}_{0:l})$ and in red $\mathbb{E}[\widehat{S}_T(\widehat{\mathbf{X}}_l)]$ with $C = 800$, $N = 70000$, $L = 50$.

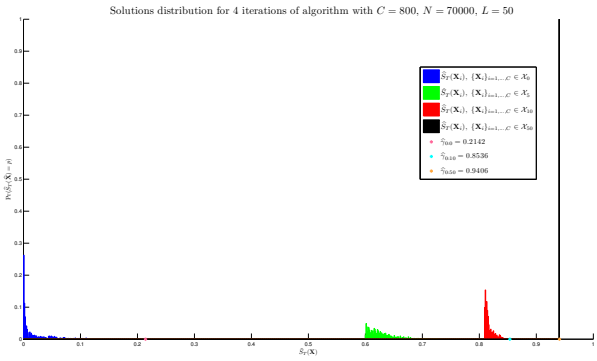
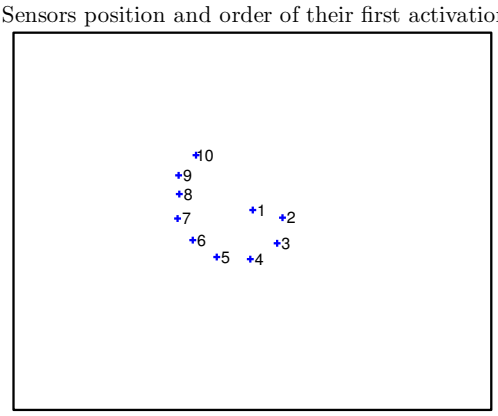


Fig. 7. Scores densities support versus iterations GSRES. In blue $l = 0$, in green $l = 5$, in red $l = 10$ and in black $l = 50$.



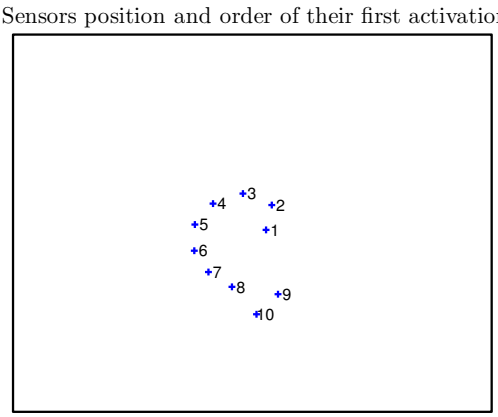


Fig. 8. Graphic of $\boldsymbol{X}^{\dagger}$: position and activation order of the 10 sensors for two other sub-optimal solutions.

nario (with CMC method for detection probability evaluation), considering the target has a myope behaviour. According to Son's researches, the best track spacing is computed thanks to the following equation:

$$TS = \max\left\{2R, \min\{\alpha TS_{ray}, TS^{\star} + \beta\}\right\}. \qquad (26)$$

TS_{ray} is considered as a good track spacing for a random tour target [1] that is approximately normally distributed, $S^{\star}$ is the largest track spacing (so-called the "furthest-on-disk" solution) and R is the sensor detection radius. In [19], α and β have been obtained by using a quasi Monte Carlo framework, more precisely, a nearly orthogonal latin hypercube (NOLH) method. After that, we have fitted an Archimedean spiral and a logarithmic spiral to our solution (see [14]). Figure 9 shows the comparison between the myope Archimedean spiral, our solution, the fitted Archimedean spiral and the fitted logarithmic spiral. We remark the sensors position and

[1] Assuming that the course change frequency is large enough versus $\frac{1}{T}$ and that the carrier speed is much larger than the target speed.

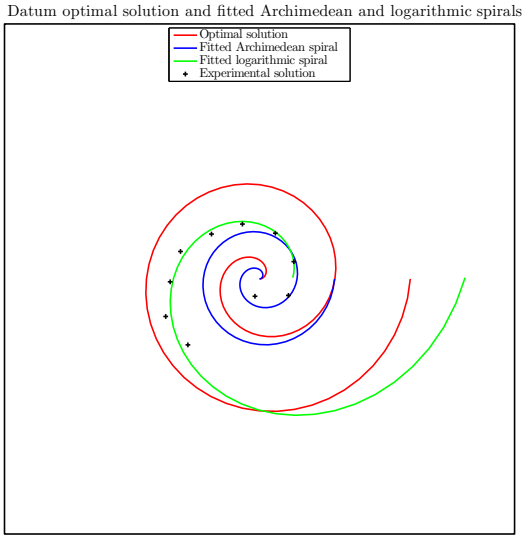


Fig. 9. Datum optimal solution and fitted Archimedean spiral

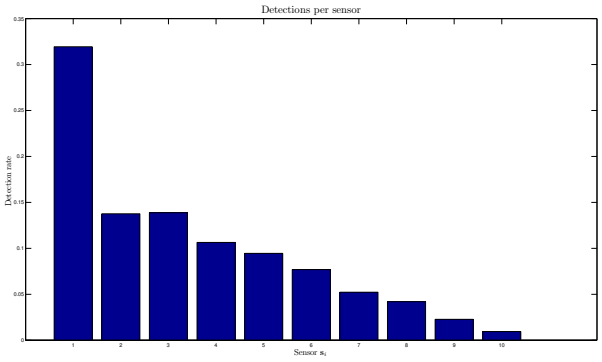


Fig. 10. Sensors detection rate for the best solution $\boldsymbol{X}^{\dagger}$.

activation rather describe a logarithmic spiral than an Archimedean spiral. This may be explained by the differences between the target dynamic models and by the fact that we don't use a single and always active sensor, but 10 sensors instead.

Figure 10 shows that the first sensor deployed detects most of the targets ($N = 70000$).

The following pictures describe how our algorithm works. Figure 11 shows the spatial distribution of the $C = 800$ solutions at initialization. We remark that for the first four sensors, the spatial distribution densities follow a Rayleigh distribution. As the carrier trajectory bounces in the search space for the next sensors, the spatial distribution densities converge toward a uniform distribution.

In figure 12 we notice that the first sensor is almost positioned. For the other sensors, some saddle points emerge.

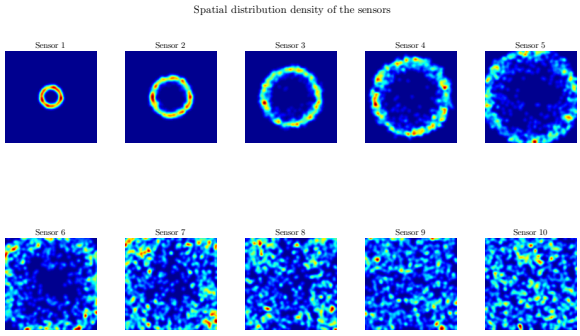


Fig. 11. Spatial sensors' pdf at initialization

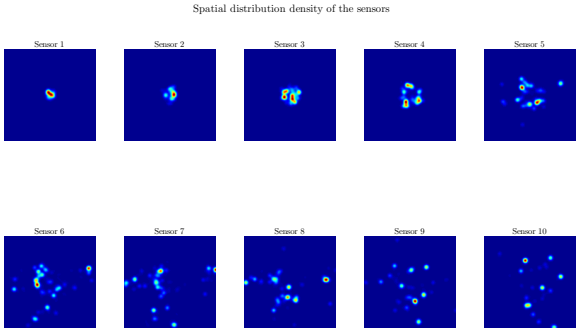


Fig. 12. Spatial sensors' pdf at iteration 5

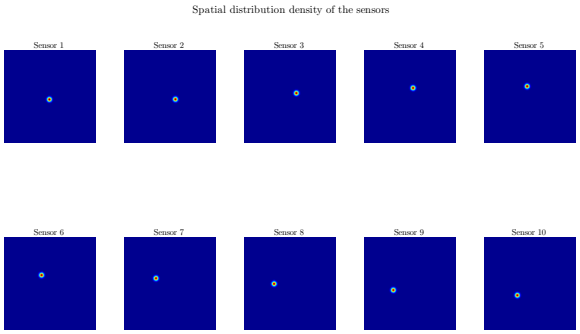


Fig. 13. Spatial sensors' pdf at final iteration

At final iteration (see figure 13), the sensors are positioned and all the spatial distribution follow a unimodal distribution.

During the Gibbs sampler's step of the optimization, we reject a new solution if it is not feasible or if its score is below γ_{l-1} at iteration l. Otherwise, we consider that the move is accepted. The two following graphics (figures 17 and 18) show the evolution of the two moves' acceptance during the optimization. The acceptance rates decrease while the optimization is conducted. The two peaks correspond to the threshold's decrease.

The next ones (figures 19 and 20) focus on the move's rejections. More precisely, the blue curves describe the rate of non-feasible solutions generated by the move, and the

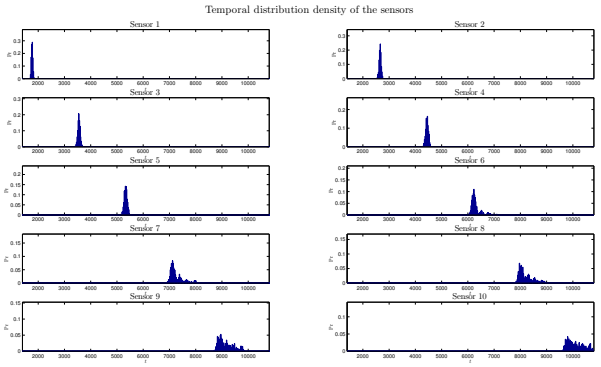


Fig. 14. Temporal sensors' pdf at initialization

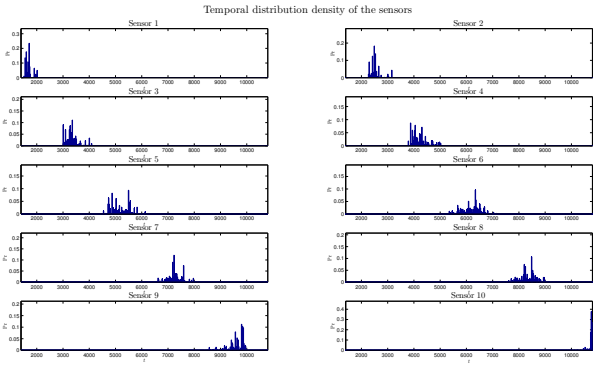


Fig. 15. Temporal sensors' pdf of sensors at iteration 5

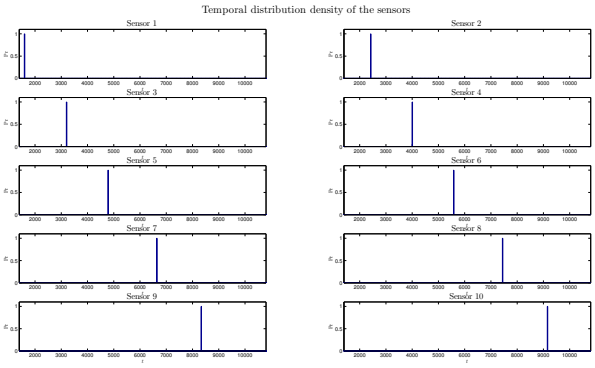


Fig. 16. Temporal sensors' pdf of sensors at final iteration

green curves describe the rate of solutions which score is below the threshold γ_{l-1} conditional on the fact they are feasible. Here again, the sharp drops of the rejections designed by the green curves correspond to the threshold decrease.

6 Conclusion and prospects

In this paper, we presented a method to compute the best deployment of sensors in order to optimize the detection probability of a smart and reactive target. Unlike existing works that use discrete constraints or only focus on the optimization of one aspect at a time, we pur-

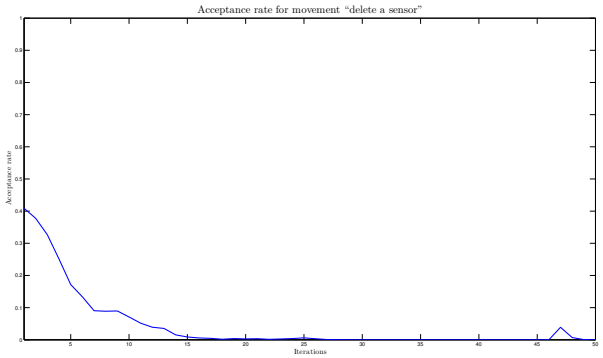


Fig. 17. Acceptance rate for move "delete a sensor"

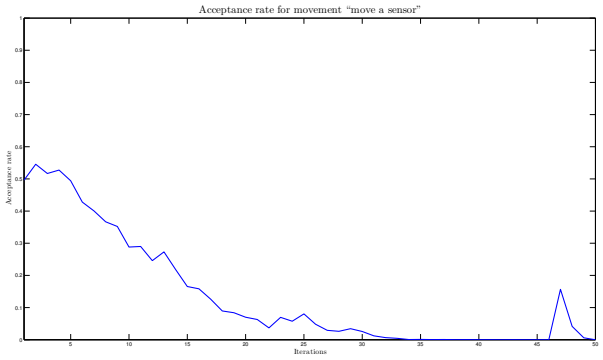


Fig. 18. Acceptance rate for move "move a sensor"

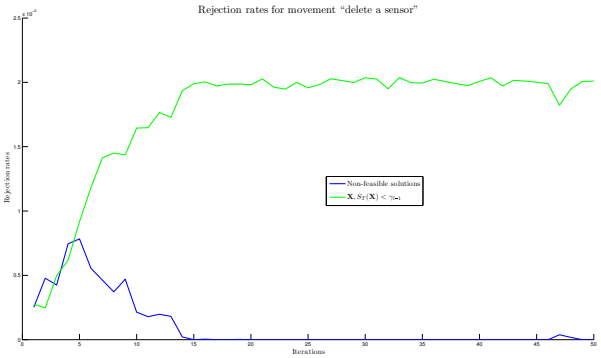


Fig. 19. Rejection rates for the move "delete a sensor"

posed to optimize both space and time with continuous constraints. This was made possible through the use of a novel stochastic optimization algorithm based on the generalized splitting method.

We tested our algorithm with the datum search problem and the results we obtained were very satisfying. In addition, these results were confirmed by existing works on much simplier cases. This demonstrate the efficiency of our dedicated Gibbs sampler and the splitting framework for this type of problem.

To reduce the rejection rate and consequently reduce the computing time, we will try to improve the efficiency of

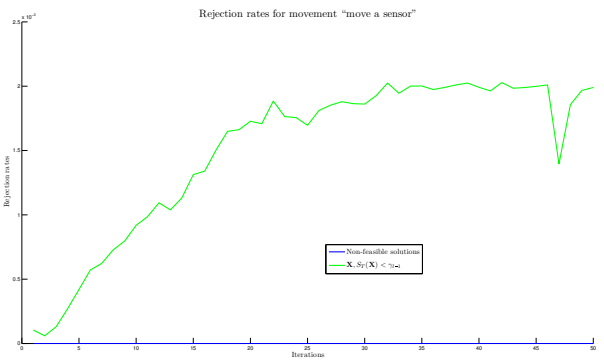


Fig. 20. Rejection rates for the move "move a sensor"

our moves. In the same time, we should focus on cooperation between sensors (*i.e. multistatic* case) and use a more complex model for sensors, signal propagation and detections (*i.e.* not cookie-cutter sensors). We also aim to take the cost of each solution into account and compute a real multiobjective optimization. To achieve this, we are currently investigating two different ways. The first one would be based on a Pareto-ranking algorithm [4] and the second would use Choquet integral for the aggregation of the user preferences [12,13].

Acknowledgement

This work was partially supported by DGA (Direction générale de l'armement). All four authors gratefully acknowledge Emile Vasta (DGA/Techniques navales) for his friendly support and interest in this work.

References

[1] Nadjib Aitsaadi, Nadjib Achir, Khaled Boussetta, and Guy Pujolle. Underwater sensor network deployment for water quality monitoring. In *ACM WUWNet'06 in conjunction with ACM MobiCom'06*, Los Angeles, California, USA, September 25 2006.

[2] Thomas Bäck and Hans-Paul Schwefel. An overview of evolutionary algorithms for parameter optimization. *Evol. Comput.*, 1:1–23, March 1993.

[3] N. Bartolini, T. Calamoneri, E. G. Fusco, A. Massini, and S. Silvestri. Snap and spread: A self-deployment algorithm for mobile sensor networks. In *DCOSS '08: Proceedings of the 4th IEEE international conference on Distributed Computing in Sensor Systems*, pages 451–456, Berlin, Heidelberg, 2008. Springer-Verlag.

[4] James Bekker and Chris Aldrich. The cross-entropy method in multi-objective optimisation: An assessment. *European Journal of Operational Research*, 211(1):112–121, 2011.

[5] Zdravko Botev and Dirk Kroese. An efficient algorithm for rare-event probability estimation, combinatorial optimization, and counting. *Methodology and Computing in Applied Probability*, 10(4):471–505, December 2008.

[6] Frédéric Cérou and Arnaud Guyader. Adaptive multilevel splitting for rare event analysis. *Stochastic Analysis and Applications*, 25(2):417–443, March 2007.

[7] Xi Chen, Hongxing Bai, Li Xia, and Yu-Chi Ho. Design of a randomly distributed sensor network for target detection. *Automatica*, 43:1713–1722, October 2007.

[8] R. F. Dell, J. N. Eagle, G. H. Alves Martins, and A. Garnier Santos. Using multiple searchers in constrained-path, moving-target search problems. *Naval Research Logistics*, 43(4):463–480, 1996.

[9] Persi Dianonis and Susan Holmes. Three examples of Monte-Carlo Markov chains. *Discrete Probability and Algorithms*, pages 43–56, 1994.

[10] D. Fudenberg and J. Tirole. *Game Theory*. MIT Press, 1991.

[11] Erik F. Golen. *Intelligent Deployment Strategies For Passive Underwater Sensor Network*. PhD thesis, Rochester Institute of Technology, May 2009.

[12] M. Grabisch. Une approche constructive de la décision multicritère. *Traitement du signal*, 22(4):321–337, 2005.

[13] M. Grabisch. L'utilisation de l'intégrale de Choquet en aide multicritère à la décision. *Newsletter of the European Working Group "Multicriteria Aid for Decision"*, 3(14):5–10, Fall 2006.

[14] Sudhanshu K. Mishra. Fitting an origin–displaced logarithmic spiral to empirical data by differential evolution method of global optimization. *The IUP Journal of Computational Mathematics*, 3(3):7–14, September 2010.

[15] James Pita, Manish Jain, Janusz Marecki, Fernando Ordóñez, Christopher Portway, Milind Tambe, Craig Western, Praveen Paruchuri, and Sarit Kraus. Deployed ARMOR protection: the application of a game theoretic model for security at the Los Angeles international airport. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems: industrial track*, AAMAS '08, pages 125–132, Estoril, 2008. International Foundation for Autonomous Agents and Multiagent Systems.

[16] James Pita, Manish Jain, Fernando Ordóñez, Milind Tambe, Sarit Kraus, and Reuma Magori-Cohen. Effective solutions for real-world Stackelberg games: when agents must deal with human uncertainties. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems - Volume 1*, AAMAS '09, pages 369–376, Budapest, 2009. International Foundation for Autonomous Agents and Multiagent Systems.

[17] X. Rong Li and V. P. Jilkov. Survey of maneuvering target tracking. Part i. Dynamic models. *Aerospace and Electronic Systems, IEEE Transactions on*, 39(4):1333–1364, 2003.

[18] Reuven Y. Rubinstein. The Gibbs cloner for combinatorial optimization, counting and sampling. *Methodology and Computing in Applied Probability*, 11(4):491–549, December 2009.

[19] Byungsoo Son. Tracking spacing for an Archimedes spiral search by a maritime patrol aircraft (MPA) in anti-submarine warfare (ASW) operations. Master's thesis, Naval Postgraduate School, December 2007.

[20] Guillaume Souris and Jean-Pierre Le Cadre. Un panorama des méthodes d'optimisation de l'effort de recherche en détection. *Traitement du Signal*, 16(6):403–424, 1999.

[21] A. Washburn. *Search and Detection*. INFORMS, 2002.

[22] A.R. Washburn. Branch and bound methods for a search problem. *Naval Research Logistics*, 45(3):243–257, 1998.